UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RICARDO DE ANDRADE MAIA

DESENVOLVIMENTO DO MICROGRAM: UM APLICATIVO PARA USO DO TELEGRAM EM RELÓGIOS COM WEAR OS

CAMPINA GRANDE - PB

2024

RICARDO DE ANDRADE MAIA

DESENVOLVIMENTO DO MICROGRAM: UM APLICATIVO PARA USO DO TELEGRAM EM RELÓGIOS COM WEAR OS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientadora : Professora Dra. Melina Mongiovi Cunha Lima Sabino

CAMPINA GRANDE - PB

2024

RICARDO DE ANDRADE MAIA

DESENVOLVIMENTO DO MICROGRAM: UM APLICATIVO PARA USO DO TELEGRAM EM RELÓGIOS COM WEAR OS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professora Dra. Melina Mongiovi Cunha Lima Sabino
Orientadora – UASC/CEEI/UFCG

Professor Dr. Adalberto Cajueiro de Farias
Examinador – UASC/CEEI/UFCG

Professor Dr. Francisco Vilar Brasileiro
Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 15 de MAIO de 2024.

CAMPINA GRANDE - PB

RESUMO

Um dos principais motivos para a concepção dos *smartwatches* e *wearables* em geral é permitir que o usuário tenha acesso rápido às informações de sua vida digital sem as distrações de utilizar um *smartphone*. Porém, a má integração da maioria dos *smartwatches* com aplicativos de mensagens instantâneas induzem o usuário a usar o telefone, limitando o potencial desses dispositivos. Diante do exposto, foi proposto o desenvolvimento de uma versão do mensageiro Telegram para o sistema Wear OS. O Telegram é o 6º mensageiro mais utilizado do mundo, de código aberto, e permite apps de terceiros. O objetivo do aplicativo é permitir ao usuário ler e responder as mensagens diretamente no *smartwatch* poupado-os das distrações de usar o *smartphone*.

DEVELOPMENT OF MICROGRAM - AN APP FOR USING TELEGRAM ON WEAR OS WATCHES

ABSTRACT

One of the main reasons for the conception of smartwatches and wearables in general is to allow users quick access to their digital life's information without the distractions of using a smartphone. However, the poor integration of most smartwatches with instant messaging apps leads users to use their phones, limiting the potential of these devices. Given this, the development of a version of the Telegram messenger for the Wear OS system was proposed. Telegram is the 6th most used messenger in the world, open source, and allows third-party apps. The goal of the app is to enable users to read and respond to messages directly on the smartwatch, saving them from the distractions of using a smartphone.

Desenvolvimento do Microgram - Um Aplicativo para Uso do Telegram em Relógios com Wear OS

Ricardo de Andrade Maia
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
ricardo.maia@ccc.ufcg.edu.br

Melina Mongiovi Cunha Lima Sabino
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
melina@computacao.ufcg.edu.br

RESUMO

Um dos principais motivos para a concepção dos *smartwatches* e *wearables* em geral é permitir que o usuário tenha acesso rápido às informações de sua vida digital sem as distrações de utilizar um *smartphone*. Porém, a má integração da maioria dos *smartwatches* com aplicativos de mensagens instantâneas induzem o usuário a usar o telefone, limitando o potencial desses dispositivos. Diante do exposto, foi proposto o desenvolvimento de uma versão do mensageiro Telegram para o sistema Wear OS. O Telegram é o 6º mensageiro mais utilizado do mundo, de código aberto, e permite apps de terceiros. O objetivo do aplicativo é permitir ao usuário ler e responder as mensagens diretamente no *smartwatch* poupado-os das distrações de usar o *smartphone*.

Keywords

smartwatches, Jetpack Compose, TDLib, mensageiros, app.

1. INTRODUÇÃO

[1]A popularidade dos *smartwatche*s aumentou rapidamente nos últimos anos. Espera-se que o tamanho do mercado de *smartwatch* em termos de volume de remessa cresça de 134,12 milhões de unidades em 2023 para 456,89 milhões de unidades em 2028 [1]. Esses aparelhos surgiram como uma forma de ponte da vida digital dos usuários com a vida real. Pequenos computadores presos ao pulso com significativo poder de processamento, permitem acesso rápido e conveniente aos dados, aplicativos e funcionalidades do *smartphone*, ao mesmo tempo em que também possibilitam a coleta de dados de saúde [2] e atividades físicas, enviando-os para o telefone e plataformas de nuvem.

Assim como o uso dos *smartphones* variou com o tempo, o foco de uso dos *smartwatches* também foi mudando. Passaram de oferecer meras replicações de notificação, extensões de aplicativos do *smartphones* e em alguns casos ligações telefônicas, para o monitoramento independente de saúde e atividades físicas e, mais recentemente, aplicativos completos, muitas vezes independentes do telefone.

Com o passar dos anos houve significativa mudança na forma que os usuários interagem com dispositivos móveis. Em particular, uma das principais mudanças foi os usuários passarem de apenas consumir conteúdo para produzir e interagir com o conteúdo. Nesse contexto, tivemos a popularização de redes sociais e mensageiros, tendo o segundo se tornado a principal forma de comunicação das pessoas [3], inclusive ultrapassando o uso de telefonia tradicional [4].

Dada esta grande mudança de paradigma na forma de comunicação das pessoas, vê-se a necessidade dos *wearables* (dispositivos vestíveis) acompanharem essa mudança de hábitos. Neste ponto a situação é bem precária, uma vez que a interação com esses mensageiros é muito pequena ou inexistente.

As soluções mais populares de mensageiros, como WhatsApp, Facebook Messenger, WeChat e Telegram [5], não oferecem aplicações complementares com *smartwatches*, e as integrações padrões dos *smartwatches*, geralmente, apenas são capazes de mostrar as últimas mensagens recebidas, levando o usuário a sempre recorrer ao telefone quando recebe uma mensagem. perdendo uma das grandes vantagens dos *smartwatches* que é a rápida disponibilidade.

Em face disto, este trabalho tem como objetivo desenvolver uma versão do aplicativo Telegram para o sistema Wear OS, o Microgram. O Telegram foi escolhido pois é um dos mensageiros mais utilizados do mundo [6, 7] e é um dos poucos que tem código aberto e permite a criação de aplicativos de terceiros. Já o Wear OS [8], por fazer parte do ecossistema Android, permite um grande alcance.

Buscou-se que o Microgram permita utilizar as principais e mais comuns funcionalidades do aplicativo, ver e responder mensagens de texto e ouvir e enviar mensagens de voz, para atender aqueles que já fazem uso do telegram em outras plataformas e entrar em um mercado bastante diverso nos *smartphones*, mas ainda pouco explorado nos *smartwatches*.

2. FUNDAMENTAÇÃO TEÓRICA

Os aplicativos móveis, são *softwares* desenvolvidos exclusivamente para dispositivos móveis como *smartphones* e

tablets. Eles se popularizaram a partir do primeiro *smartphone*, em 2007, com o sistema operacional iOS. Hoje já estão em outros dispositivos como *smartTVs* e *wearables*, a exemplo dos *smartwatches*. O objetivo deles é disponibilizar funcionalidades de forma fácil e intuitiva [9].

Estes aplicativos executam em um sistema operacional. Nos *smartwatches* um dos sistemas operacionais utilizado é o Wear OS. Os *smartwatches* são relógios inteligentes, semelhantes aos relógios de pulso tradicionais, mas que disponibilizam outras funcionalidades além de marcar a hora, por exemplo o envio e recebimento de mensagens, conexão com o *smartphone* para o recebimento de notificações, chamadas de voz e o monitoramento de saúde e atividades físicas, muitas vezes eles possuem tela sensível ao toque.

O Wear OS é o sistema da Google para *smartwatches* que é usado tanto em dispositivos da marca quanto no de empresas parceiras. Esse sistema é construído com base no Android (portanto se trata de um sistema Linux) e oferece um alto nível de compatibilidade com outros aparelhos que usam o Android [8].

Os *smartwatches* vêm recebendo vários tipos de aplicativos independentes dos *smartphones*. Recentemente, em julho de 2023, a empresa Meta lançou a primeira versão oficial do WhatsApp para Wear OS 3. O aplicativo no *smartwatch* permite responder mensagens de texto e voz, além de enviar mensagens rápidas e emojis.

Outro aplicativo de mensagens que se adiantou no mercado dos *smartwatches* foi o Line. De origem japonesa, e bastante popular em algumas regiões da Ásia, o aplicativo não ganhou popularidade no Brasil como outros citados anteriormente, porém sua versão para Wear OS é capaz de visualizar mensagens, enviar textos, emojis e mensagens de voz.

Como um indicativo do crescimento do interesse do mercado em aplicativos para dispositivos vestíveis, a Big Tech Google também tem lançado versões de seus aplicativos, como Gmail e Google Agenda, para o Wear OS.

O Telegram é um popular concorrente dos mensageiros citados anteriormente, que atualmente não possui versão oficial para o Wear OS (houve uma versão para o Android Wear 2 que foi descontinuada e está indisponível). Ele é um serviço de mensagem baseado em nuvem unificado, com foco em privacidade, segurança e velocidade. Este serviço não disponibiliza publicamente o código do servidor, nem permite que servidores de terceiros se conectem a ele, mas a criptografia e a API (Application Programming Interface ou Interface de Programação de Aplicação) usadas no servidor do Telegram são completamente documentadas e abertas. Isso possibilita que terceiros criem seus próprios aplicativos clientes e se conectem ao serviço de nuvem do Telegram. Logo, é possível criar um aplicativo cliente também para o Wear OS.

Como é o sexto aplicativo mais utilizado e mais baixado do mundo [6], existe uma grande comunidade que tem preferência pelo Telegram como mensageiro, de forma que o Microgram herda os diferenciais do Telegram que atraem essa comunidade, como o foco em privacidade em que os dados dos usuários não são coletados para marketing, nem vendidos para terceiros.

Para o desenvolvimento de um aplicativo cliente do Telegram para o sistema Wear OS é necessário o conhecimento de algumas tecnologias, sendo elas:

- TDLib - *Telegram Database Library* é um cliente Telegram multiplataforma e totalmente funcional. Projetado para ajudar desenvolvedores terceirizados a criar seus próprios aplicativos personalizados usando a plataforma Telegram [10]. Ele gerencia vários detalhes da implementação de rede, criptografia e armazenamento local de dados [11].
- SQLite - uma biblioteca em linguagem C que implementa um mecanismo de banco de dados SQL pequeno, rápido, independente, de alta confiabilidade e completo. Ele está integrado em todos os telefones celulares e na maioria dos computadores e vem integrado em inúmeros outros aplicativos que as pessoas usam todos os dias. O código-fonte SQLite é de domínio público e pode ser usado gratuitamente por todos para qualquer finalidade [12].
- Docker - uma plataforma de código aberto para desenvolvimento, envio e execução de aplicativos dentro de *containers* virtuais. Um *container* é um processo em ambiente restrito, executado em uma máquina *host*, isolado de todos os outros processos em execução nessa máquina *host*, usando um sistema de arquivos. Este sistema de arquivos isolado é fornecido por uma imagem, e a imagem deve conter tudo o que é necessário para executar um aplicativo - todas as dependências, configurações, *scripts*, binários, etc. O *container* pode ser executado em máquinas locais, máquinas virtuais ou implantado na nuvem, além de ser executado em qualquer sistema operacional, está isolado de outros *containers* e executa seu próprio *software*, binários, configurações, etc [13].
- JNI - a interface nativa Java. Ela define uma maneira para o *bytecode* que o Android compila, a partir de código escrito nas linguagens de programação Java ou Kotlin, interagir com o código nativo (escrito em C/C++). A JNI é neutra em termos de fornecedor, tem suporte para carregar código de bibliotecas compartilhadas dinâmicas [14].
- Jetpack Compose - “ um kit de ferramentas moderno do Android para criar IUs nativas” [15]. “Ele simplifica e acelera o desenvolvimento da interface, incluindo a adaptação a qualquer formato, de *smartphones*, dobráveis e tablets a TVs e *wearables*” [16].
- SDK Android - é um kit de desenvolvimento de *software* para o ecossistema de *software* Android que engloba um conjunto abrangente de ferramentas de desenvolvimento. Estas incluem um depurador, bibliotecas, um emulador baseado em QEMU, documentação, código de exemplo e tutoriais. O SDK faz parte da IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado) oficial do Android Studio, mas suas ferramentas e recursos podem ser usados de forma independente [17].

Por fim, no desenvolvimento do aplicativo deste trabalho, a arquitetura de *software* escolhida foi a MVC (*Model-View-Controller*).

“A arquitetura MVC foi criada nos anos 80 na Xerox Parc, por Trygve Reenskaug, que iniciou em 1979 o que viria a ser o nascimento do padrão de projeto MVC” [18].

Segundo Daoudi *et al*. [19] o padrão busca a separação entre o modelo de negócio e a camada de apresentação. Para isso define três camadas principais: o Modelo (*Model*) que representa o conjunto de informações da aplicação e as regras para lidar com essa informação, a Visão (*View*) que provê uma representação visual do Modelo e pode alterar o estado do modelo através de mensagens. Por último, o Controlador (*Controller*) que realiza a comunicação entre o Modelo e a Visão, controlando a Visão e traduzindo as interações do usuário para o Modelo.

Na Figura 1, observa-se as três camadas da Arquitetura MVC e as interações entre elas. O *Controller* faz toda a comunicação entre a View e o *Model*, que não interagem diretamente entre si. Essa decisão arquitetural permite isolar a interface da lógica da aplicação e, dessa forma, possibilita que a aplicação utilize diferentes interfaces sem a necessidade de alterar o modelo.

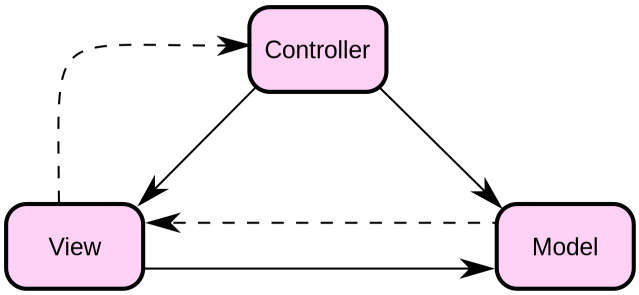

Figura 1 - Arquitetura MVC [18]

3. SOLUÇÃO

3.1 Solução Proposta

Propõe-se o desenvolvimento do Microgram, uma versão do Telegram para os dispositivos com o sistema Wear OS. A escolha desse sistema se justifica por ser encontrado em vários modelos do mercado, por sua familiaridade para aqueles que já fazem uso do sistema Android e por sua compatibilidade com outros dispositivos que utilizam este sistema.

Buscou-se que o Microgram possibilite uma experiência menos limitada do que apenas o recebimento de notificações e envio de mensagens prontas, como é comum em muitos aplicativos atuais, permitindo o envio e recebimento de mensagens por texto e por voz, listagem de conversas, visualização de conversas com um contato, principalmente, sem a dependência do *smartphone*. Para tanto, foi desenvolvido um aplicativo com uma interface adaptada ao tamanho da tela de um *smartwatch*, para oferecer uma experiência mais agradável e fluida no dispositivo. Este se conecta de maneira eficiente com os serviços do Telegram, através do TDLib.

O código fonte do aplicativo está disponível em um repositório público no Github sob a licença MIT. Esta licença permite que qualquer pessoa faça uso do código fonte, modifique ou distribuía, desde que seja dado os devidos direitos autorais e se mantenha os arquivos da licença. Além disso, o autor do código não dá garantias nem se responsabiliza por danos decorrentes de seu uso.

Futuramente, o Microgram estará disponível na Google Play e em seu próprio *website* para *download* do APK.

3.2 Funcionalidades

A primeira versão do Microgram possui 6 funcionalidades, distribuídas em 5 telas.

3.2.1 Login

No primeiro acesso do usuário ao aplicativo é exibida uma tela de boas vindas, seguida por uma tela com instruções e um QR Code para que o usuário possa fazer login de sua conta Telegram (Figura 2). Essas telas só são exibidas no primeiro acesso, devendo o usuário escanear o QR Code com o Telegram do *smartphone*, através da função de conectar novo dispositivo, para realizar o *login* no *smartwatch*.

Figura 2 - Tela de boas vindas e *login*

3.2.2 Listagem de Conversas

Após efetuar o *login*, e todas as vezes que abrir o aplicativo nos acessos seguintes, o usuário é direcionado para a tela de listagem de conversas. Nessa tela, o usuário visualiza as conversas em ordem cronológica desde a última conversa ativa, e pode rolar a tela para visualizar as conversas mais antigas. Cada item da lista mostra uma foto do contato, o nome e a última mensagem trocada naquela conversa. No final da lista há um botão para carregar mais conversas, como pode ser visto na Figura 3.

3.2.3 Listagem de Mensagens de uma Conversa

Quando o usuário clica em uma das conversas, o aplicativo abre a tela que permite visualizar todas as mensagens trocadas com aquele contato, também em ordem cronológica a partir da última mensagem (Figura 4). Além disso, existem 2 botões nessa tela. O primeiro abre a tela para digitar mensagens de texto, e o segundo abre a tela onde é possível gravar áudio.

3.2.4 Envio de Mensagens

Finalmente, na tela para digitar mensagens de texto estão o teclado virtual, uma pré-visualização do que foi digitado e o botão de enviar (Figura 5). Enquanto que, na tela de gravação de áudio (Figura 6), estão o cronômetro mostrando o tempo do áudio e um botão para enviar.

As imagens das telas do aplicativo foram tiradas do protótipo do Microgram feito no *software* Figma.

Figura 3 - Tela de listagem de conversas

Figura 4 - Tela de visualização de conversas

Figura 5 - Tela para digitar mensagem de texto

Figura 6 - Tela para gravação de áudio

3.3 Arquitetura

3.3.1 Servidor do Telegram

O código do servidor do Telegram não é aberto, nem permite que servidores de terceiros se conectem a ele. Logo o aplicativo cliente precisa se conectar diretamente ao servidor em nuvem do Telegram e não a um *backend* próprio. Para que o Microgram conecte-se ao serviço de nuvem do Telegram foi preciso realizar um cadastro no Telegram para obter um api_id e um api_hash, que permitem a autenticação, e seguir os Termos de Serviço da API (API Terms of Service) [20].

3.3.2 Mobile

Existem duas bibliotecas para desenvolver aplicativos para Wear OS. A primeira é baseada em visualizações, enquanto que a segunda usa o Jetpack Compose para Wear OS. No caso do Microgram decidiu-se utilizar a segunda alternativa juntamente com a linguagem Kotlin, pois é uma recomendação da Google. As vantagens de utilizar o Jetpack Compose incluem: menos código (uma vez que dispensa o uso de arquivos XML), eficiência, intuitividade (utiliza uma API declarativa) e acelera o desenvolvimento, pois é compatível com código já existente.

Para se conectar com o servidor em nuvem do Telegram o Microgram utiliza o TDLib como uma dependência do aplicativo. Ele gerencia vários detalhes da implementação de rede, criptografia e armazenamento local de dados [11]. No caso do armazenamento de dados, o TDLib faz uso do SQLite para realizar cache dos dados da aplicação, por exemplo, dados de conversas e mensagens, permitindo que o usuário possa acessá-las mesmo offline.

Sendo o TDLib desenvolvido em C++ e o código Kotlin executado no Android Runtime, uma implementação da JVM (*Java Virtual Machine* ou Máquina Virtual Java) para o Android, tornou-se necessário também utilizar o JNI para carregar o código nativo C++ e permitir a interoperabilidade entre as duas linguagens.

3.3.3 Estrutura do Projeto

A estrutura do projeto (Figura 7) foi escolhida tendo como base a estrutura gerada pelo Android Studio, a arquitetura MVC e experiências com projetos anteriores.

Na pasta *telegram* estão os pacotes relacionados à interação com o TDLib. O primeiro pacote *controllers* possui as entidades que interagem diretamente com o TDLib executando chamadas às funções do TDLib e definindo os *handlers* correspondentes para *callback* dessas chamadas. Os *handlers* são os responsáveis por monitorar e reagir as respostas às chamadas do TDLib, incluindo as respostas do servidor, e delegar o tratamento dos dados ao *service* correspondente. Por último, os *services* são os responsáveis pela lógica de negócio, autenticação e por atualizar os estados da aplicação.

A pasta *model* contém classes que representam dados do aplicativo, por exemplo conversas e usuários.

Em *presentation*, o pacote *theme* contém os arquivos para a estilização do aplicativo. Já o pacote *ui* possui os arquivos das telas do aplicativo e dos estados. Por fim, a classe MainActivity é o ponto de partida da aplicação que configura a interface e o ambiente de execução.

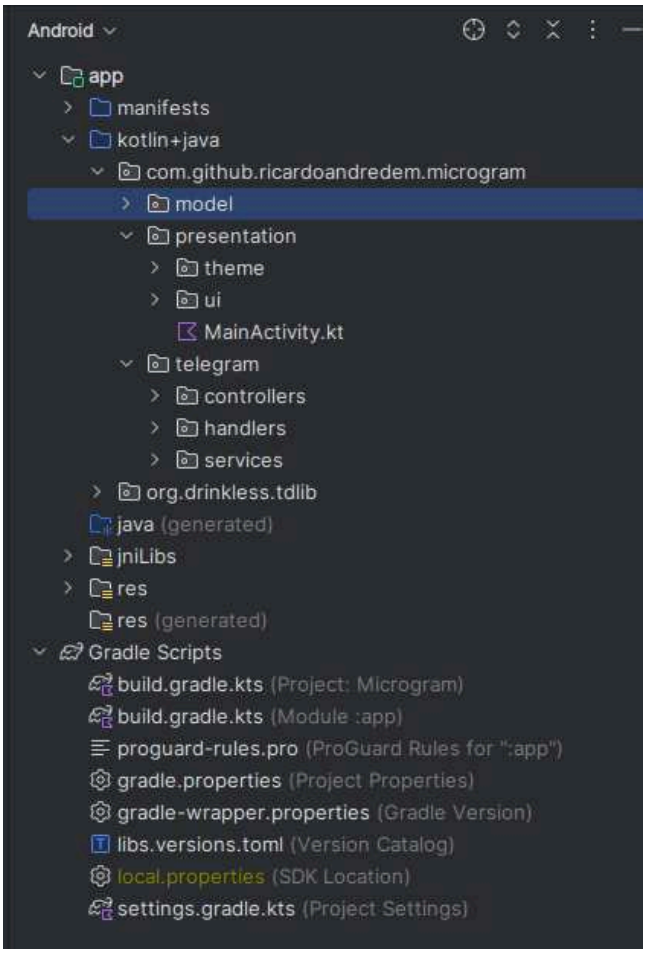

Figura 7 - Estrutura do projeto

4. SISTEMA EM USO

Como o MVP do aplicativo não ficaria pronto em tempo hábil para a avaliação com usuários, foi feita uma avaliação preliminar do aplicativo a partir do protótipo de alta fidelidade no Figma.

Para coletar dados referentes a experiência do usuário com o protótipo foi distribuído um formulário de avaliação. Este formulário é uma adaptação do Questionário de Usabilidade de Sistema de Computador (Computer System Usability Questionnaire - CSUQ) [21]. Baseado no *IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use,* o questionário foi adaptado para avaliar o Microgram de acordo com os seguintes aspectos: facilidade de uso, clareza das informações, interface e *design*, funcionalidades e capacidades e satisfação geral.

No formulário [22] foi pedido que o usuário utilizasse livremente o protótipo, interagindo com a listagem de conversas, com a listagem de mensagens de uma conversa e com as telas de envio de mensagens. Após essa interação os participantes responderam uma lista de perguntas na qual eles deveriam indicar o seu grau de concordância com as afirmações apresentadas, cujas respostas variam em uma escala linear de 1 a 5, em que 1 indica discordância total e 5 concordância total. No final os participantes deveriam listar pontos positivos e negativos da aplicação.

Obteve-se 11 resultados nessa pesquisa. Pelos resultados verificou-se que os usuários ficaram bastante satisfeitos com o Microgram. Em especial foi elogiada a interface bonita, minimalista, de uso fácil, prático e útil.

As maiores ressalvas feitas são sobre a necessidade de mais funcionalidades, como ligações, visualizar o perfil de um contato e poder cancelar mensagens de voz antes de serem enviadas.

Assim, fica indicado que o Microgram, como proposto, foi bem aceito. Apesar de ainda carecer de mais funcionalidades, o Microgram mostrou-se um projeto promissor.

5. EXPERIÊNCIAS E LIÇÕES APRENDIDAS

5.1 Processo de desenvolvimento

Primeiramente foi feito um estudo para a escolha das tecnologias que poderiam ajudar no processo de desenvolvimento. Neste estudo, identificou-se duas bibliotecas muito úteis e que facilitam partes mais complexas da implementação, o TDLib e uma biblioteca para geração do QRCode necessário na autenticação do usuário.

Em seguida, houve uma revisão de algumas tecnologias, principalmente do Android, pois já havia um conhecimento anterior deste, em versões antigas ainda utilizando a linguagem Java. Contudo, como o SDK do Android passou por atualização para adicionar novas tecnologias, a exemplo da linguagem Kotlin e do *framework* Jetpack Compose, foi preciso fazer uma revisão dos conhecimentos, incluindo estas novas tecnologias necessárias para o desenvolvimento no Wear OS.

Em paralelo, houve o estudo das tecnologias utilizadas para desenvolver o aplicativo com as quais não havia experiência anterior, como Docker e TDLib.

No caso do TDLib, como a documentação da biblioteca é bastante reduzida, foi necessária a leitura do código fonte do TDLib, bem como o código fonte de outros aplicativos clientes do Telegram, sendo um deles a versão *web* do Telegram feita em React [23]. Também foi necessário aprender alguns tópicos avançados de Java (JNI e Java Reflection), e a compilação de projetos C e Docker, porque o TDLib deve ser compilado para ser adicionado como dependência em um projeto.

Para a análise e planejamento dos requisitos mínimos do MVP (*Minimum Viable Product* ou Produto Viável Mínimo) do Microgram e da interface de usuário, inicialmente, distribuiu-se um formulário [24] que tinha como objetivo avaliar o perfil de uso do Telegram dos participantes, quais funcionalidades eram as mais desejadas em uma versão do Telegram para Wear OS, quais as preferências de interface e problemas e desafios existentes no uso do Telegram. Este formulário obteve 27 respostas.

Quanto às funcionalidades, a troca de mensagens diretas, grupos de conversas, chamadas de voz e canais foram as mais votadas (Figura 8). Devido ao tempo disponível para o desenvolvimento do MVP e considerando a finalidade básica de um aplicativo de mensagens, foram escolhidas as funcionalidades de troca de mensagens por texto e de mensagens por voz para a primeira versão do aplicativo.

Já em relação a interface, a maior porcentagem dos participantes escolheram a interface do tipo lista de conversas em que se pode rolar verticalmente para selecionar e visualizar as mensagens de cada conversa, sendo seguida pela interface simplificada e minimalista com funções essenciais (Figura 9). Muitos também apontaram o desejo de ser possível personalizar a aparência da interface (Figura 10), contudo, para o MVP, decidiu-se implementar uma interface do tipo lista, com poucos elementos, deixando a possibilidade de personalização para atualizações futuras.

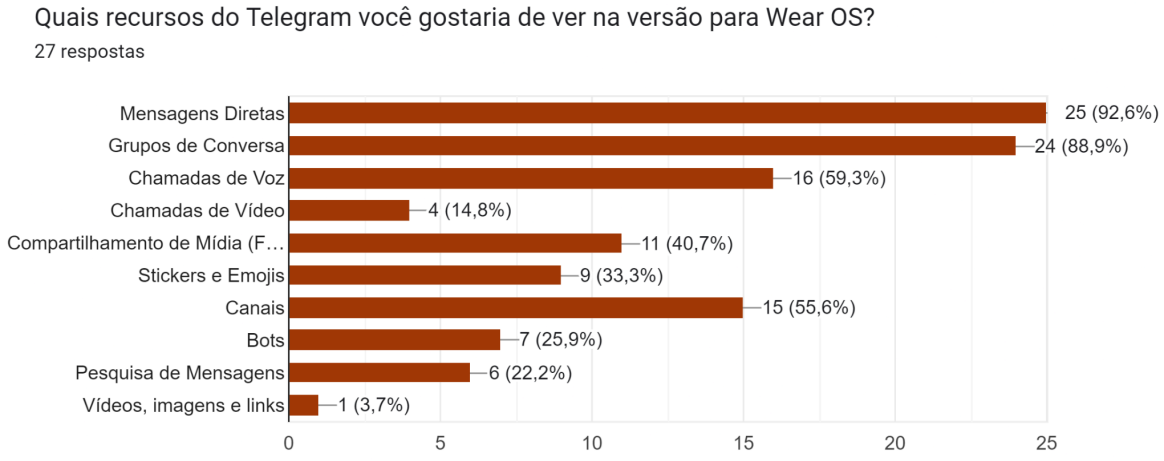

Figura 8 - Gráfico com as respostas ao questionamento sobre recursos desejados no Microgram

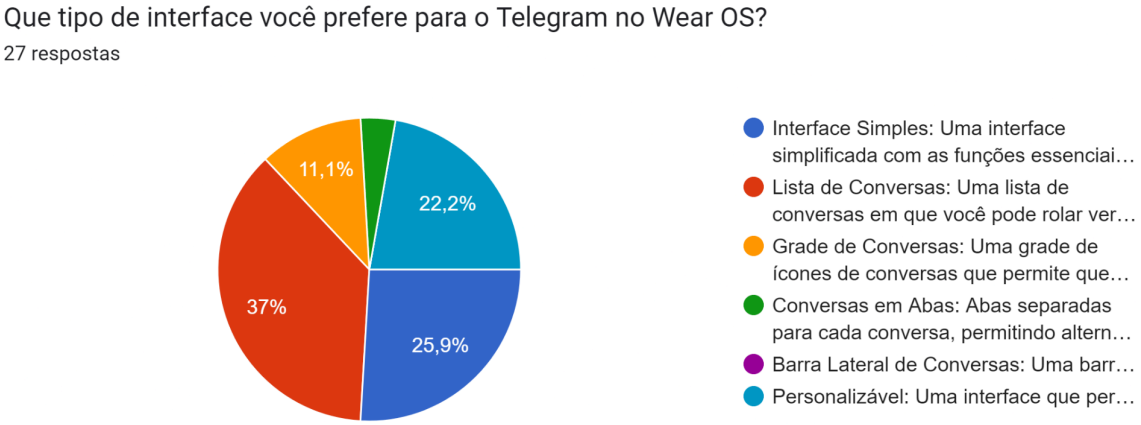

Figura 9 - Gráfico com as respostas ao questionamento sobre o tipo de interface desejada para o Microgram

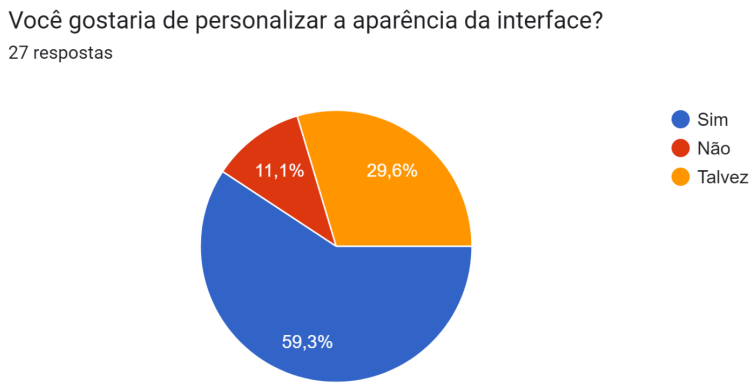

Figura 10 - Gráfico com as respostas ao questionamento sobre a personalização da aparência do Microgram

Existem ainda funcionalidades difíceis de serem adaptadas ou de pouca utilidade no *smartwatch*, como tirar fotos e fazer vídeo chamadas.

Continuando, foi feito um estudo da interface do Telegram e da estrutura comum de aplicativos de *smartwatch*, com o objetivo de melhor embasar a adaptação da interface do Telegram. Nesse processo procurou-se identificar os elementos de interface e funcionalidades essenciais e mais solicitados pelos usuários, por meio do formulário, e adequá-los para as pequenas telas dos *smartwatches*.

Este estudo foi realizado não só pelo tamanho da tela, como pelas capacidades reduzidas de um *smartwatch*, não sendo possível fazer um porte do Telegram com todas as suas funcionalidades.

Ademais, o aplicativo não visa ser um substituto da versão do *smartphone*, mas sim um aplicativo complementar que permite executar as ações mais essenciais do aplicativo, diminuindo a quantidade de vezes que os usuários precisam recorrer ao *smartphone* para uma rápida troca de mensagens.

Em seguida à análise da interface, criou-se um protótipo de alta fidelidade do MVP no *software* Figma.

Após a prototipação, a implementação do código foi feita em *sprints* (período curto e fixo de tempo) para dividir o desenvolvimento em ciclos. E para gerenciar as tarefas foi utilizado o Kanban, dividindo as atividades de desenvolvimento, estudo e pesquisa em cartões e organizando-os em listas de pendente, em andamento e concluído. Esta abordagem permitiu uma visão razoável do fluxo de trabalho apesar dos atrasos.

5.2 Desafios

A primeira dificuldade encontrada consistiu no fato de o TDLib ter pouquíssima documentação, pois grande parte de sua documentação disponível é composta de instruções para a compilação, tornando necessária a leitura do código fonte da biblioteca e de outros projetos clientes do Telegram para compreender como o TDLib e o próprio Telegram funcionam.

Como o TDLib deve passar por uma compilação prévia para uso como dependência no projeto de novos aplicativos, a própria compilação foi um desafio. Isso porque ela é feita utilizando versões específicas do Ubuntu com uma grande quantidade de dependências que precisam ser instaladas e uma compilação complexa em C++, mesmo utilizando o CMake (sistema para geração automatizada).

No entanto, durante o desenvolvimento do Microgram a biblioteca TDLib foi atualizada, sendo adicionado um arquivo dockerfile, o que facilitou recompilações, mas adicionou a necessidade aprender Docker e de adicionar um complemento ao Docker, o docker-buildx, pois o comando docker build indicado na documentação do TDLib está desatualizado e não funciona. O TDLib também possui elevado tempo de compilação e altos requisitos computacionais.

Apesar de todas as dificuldades, o TDLib cuida de vários detalhes de implementação de rede, criptografia e armazenamento local de dados, o que do contrário tornaria o desenvolvimento do aplicativo muito mais complexo e oneroso, justificando o uso da biblioteca.

A pouca experiência com o desenvolvimento Android, em versões mais antigas, tornou necessário destinar um tempo para complementar e atualizar os conhecimentos em Android e aprender várias tecnologias novas adicionadas ao SDK do Android, importantes para o projeto, como Jetpack Compese e a linguagem Kotlin. Também para aprender JNI e a carregar código nativo C++ em um projeto com outra linguagem (Kotlin) que executa na máquina virtual ART (Android Runtime). Todo esse processo levou mais tempo do que o esperado, tendo em vista que o tempo para aprender a linguagem Kotlin e o Jetpack Compose foi subestimado.

Recriar a interface do aplicativo para um *smartwatch* representou outra dificuldade, pois apesar do Wear OS ser uma versão do Android para dispositivos vestíveis, o *design* de interfaces para estes sistemas é bastante diferente. Enquanto as versões para *smartphones* utilizam interfaces ricas, complexas, com vários elementos e com várias formas de interações, as versões para *smartwatches* precisam ter poucos elementos, de fácil visualização e botões grandes com interação simples. Desta forma, foram feitas muitas pesquisas sobre versões de *smartwatch* de outros aplicativos e aplicativos semelhantes, como o de mensagens SMS, e estudo de guias de princípios de *design* para *smartwatches*.

6. TRABALHOS FUTUROS

Em face do crescimento do mercado de *smartwatches* e da necessidade cada vez maior de aplicativos para estes dispositivos, sendo a variedade de aplicativos mensageiros, independentes do *smartphone*, ainda bastante limitada, propôs-se o desenvolvimento do Microgram, um aplicativo cliente do Telegram para o Wear OS.

De acordo com as avaliações realizadas, o Microgram foi bem aceito, apesar de precisar de mais funcionalidades para atender os requisitos dos usuários.

Novas funcionalidades e melhorias que poderão ser adicionadas no futuro são: implementação de ligação por voz, visualizar informações de perfil do usuários e dos contatos, análise de desempenho do aplicativo e otimização, suporte ao envio de stickers, suporte a visualização de mensagens multimídias (imagens, vídeos, *stickers*, *stickers* animados e emojis animados), traduzir o aplicativo para outros idiomas, além da personalização de interface indicada pelos participantes do formulário de requisitos.

7. REFERÊNCIAS

[1] MordorIntelligence. 2023. Tamanho do mercado de smartwatches e análise de ações – Tendências e previsões de crescimento (2024 – 2029). Disponível em: https://www.mordorintelligence.com/pt/industry-reports/smartwatch-market.

[2] Almeida, S. M.. A maturidade do mercado através da evolução tecnológica: o caso dos Smartwatches. Dissertação de Mestrado, Faculdade de Economia do Porto, Porto, Portugal, 2017.

[3] g1. 2022. Brasileiros são os que passam mais tempo por dia no celular, diz levantamento. Disponível em: https://g1.globo.com/tecnologia/noticia/2022/01/12/brasileiros-sao-os-que-passam-mais-tempo-por-dia-no-celular-diz-levantamento.ghtml. .

[4] Komo, A. E. Eri-chain: aplicativo de troca de mensagens instantâneas factível de auditorias e distribuído. Trabalho de Conclusão de Curso, Escola Politécnica da Universidade de São Paulo, São Paulo, 2018.

[5] Statista. 2024. Most popular global mobile messenger apps as of April 2024, based on number of monthly active users. Disponível em: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/.

[6] Du Rove's Channel. 2024. Telegram reached 900 million monthly users. Disponível em: https://t.me/durov/260.

[7] Telegram. 2024. Meu Perfil, Canais Recomendados e Mais 15 Recursos - 900 Milhões de Usuários. Disponível em: https://telegram.org/blog/my-profile-and-15-more/pt-br#900-milhoes-de-usuarios.

[8] Syozi, R., Ciriaco, D. Canatech. 2024. O que é Wear OS? Disponível em: https://canaltech.com.br/apps/o-que-e-wear-os/.

[9] Bocard, T. Use. 2021. O que são aplicativos? Definição da desenvolvedora Usemobile. Disponível em: https://usemobile.com.br/aplicativo-movel/.

[10] Telegram. 2024. Telegram Database Library. Disponível em: https://core.telegram.org/tdlib.

[11] Telegram. 2018. TDLib – Build Your Own Telegram. Disponível em: https://telegram.org/blog/tdlib.

[12] SQLite. 2024. What Is SQLite? Disponível em: https://www.sqlite.org/.

[13] docker.docs. 2024. Overview of the get started guide. Disponível em: https://docs.docker.com/get-started/.

[14] Android Developers. 2024. JNI tips. Disponível em: https://developer.android.com/training/articles/perf-jni.

[15] Android Developers. 2024. Why adopt Compose. Disponível em: https://developer.android.com/develop/ui/compose/why-adopt.

[16] Android Developers. 2024. Develop UI for Android. Disponível em: https://developer.android.com/develop/ui.

[17] Wikipédia. 2024. Android SDK. Disponível em: https://en.wikipedia.org/wiki/Android_SDK.

[18] Wikipédia. 2023. MVC. Disponível em: https://pt.wikipedia.org/wiki/MVC.

[19] Daoudi, A., *et al*. 2019. An exploratory study of MVC-based architectural patterns in Android apps. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19). Association for Computing Machinery, New York, NY, USA, 1711–1720. https://doi.org/10.1145/3297280.3297447.

[20] Telegram. 2024. Creating your Telegram Application. Disponível em: https://core.telegram.org/api/obtaining_api_id.

[21] Perlman, G. Computer System Usability Questionnaire. Disponível em: https://garyperlman.com/quest/quest.cgi.

[22] Maia, R. A. 2024. Avaliação do Microgram - Um cliente do Telegram para Wear OS. Disponível em: https://forms.gle/sf5QM3s5fyTBbuoR6.

[23] Nadymov, E. 2021. telegram-react. Disponível em: https://github.com/evgeny-nadymov/telegram-react.

[24] Maia, R. A. 2023. Pesquisa de Requisitos para o Telegram no Wear OS. Disponível em: https://forms.gle/2hT6jHcH498p5zGFA.